# Install necessary libraries:

# pip install networkx python-louvain matplotlib

import networkx as nx

import community.community\_louvain as community

import matplotlib.pyplot as plt

def detect\_habitat\_clusters():

    """

    Simulates a habitat network using NetworkX and applies the Louvain

    community detection algorithm to find densely connected habitat clusters.

    In this simulation:

    - Nodes are individual habitat patches (A, B, C...).

    - Edges are potential wildlife corridors.

    - Edge weights represent the 'connectivity score' (higher weight = better corridor).

    """

    print("--- Starting Habitat Cluster Detection ---")

    # 1. Simulate the Habitat Network Data

    # Nodes: Habitat Patches (A through K)

    # Edges: (Patch1, Patch2, Connectivity\_Score)

    # Note: A higher score indicates a stronger, easier connection (e.g., a low-resistance corridor).

    habitat\_edges = [

        ('A', 'B', 0.9), ('A', 'C', 0.8), ('B', 'C', 0.7),

        ('A', 'D', 0.1),  # Weak link between A/D

        ('D', 'E', 0.95), ('D', 'F', 0.85), ('E', 'F', 0.9),

        ('G', 'H', 0.98), ('G', 'I', 0.92), ('H', 'I', 0.88),

        ('C', 'E', 0.2),  # Weak link connecting Group 1 and Group 2

        ('F', 'G', 0.15), # Weak link connecting Group 2 and Group 3

        ('J', 'K', 0.99), # Isolated mini-cluster

        ('I', 'J', 0.1), # Very weak link to isolated cluster

    ]

    # 2. Create the Graph

    G = nx.Graph()

    # Add weighted edges

    G.add\_weighted\_edges\_from(habitat\_edges, weight='weight')

    print(f"Network created with {G.number\_of\_nodes()} habitat patches and {G.number\_of\_edges()} corridors.")

    # 3. Apply Louvain Community Detection

    # The Louvain method partitions the graph into communities such that the

    # modularity within each community is maximized.

    partition = community.best\_partition(G, weight='weight')

    # Calculate Modularity Score: A metric to assess the quality of the partition.

    # Scores closer to 1 indicate a strong community structure.

    modularity = community.modularity(partition, G, weight='weight')

    print(f"\nLouvain Algorithm Results:")

    print(f"Identified {max(partition.values()) + 1} distinct habitat clusters.")

    print(f"Modularity Score: {modularity:.4f} (A high score suggests a strong clustering structure).")

    # Map node to its cluster ID (Community ID)

    cluster\_mapping = {node: cluster\_id for node, cluster\_id in partition.items()}

    print("\n--- Habitat Cluster Mapping ---")

    for node, cluster\_id in sorted(cluster\_mapping.items()):

        print(f"Patch {node}: Cluster {cluster\_id}")

    # 4. Visualization

    plt.figure(figsize=(10, 7))

    # Get the color map based on the partition (each cluster gets a unique color)

    cmap = plt.cm.get\_cmap('viridis', max(partition.values()) + 1)

    # Draw the graph

    # Set the layout for consistent positioning

    pos = nx.spring\_layout(G, seed=42)

    # Draw nodes, colored by their assigned community (cluster)

    nx.draw\_networkx\_nodes(

        G,

        pos,

        partition.keys(),

        node\_size=1200,

        cmap=cmap,

        node\_color=list(partition.values())

    )

    # Draw edges with thickness based on connectivity score (weight)

    all\_weights = [G[u][v]['weight'] for u, v in G.edges()]

    nx.draw\_networkx\_edges(

        G,

        pos,

        width=[w \* 5 for w in all\_weights], # Scale width for visibility

        alpha=0.6,

        edge\_color='gray'

    )

    # Draw labels

    nx.draw\_networkx\_labels(G, pos, font\_color='white', font\_weight='bold')

    plt.title(f"Habitat Clusters Identified by Louvain Algorithm\nModularity: {modularity:.4f}", fontsize=14)

    plt.axis('off')

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    detect\_habitat\_clusters() output

--- Starting Habitat Cluster Detection ---

Network created with 11 habitat patches and 14 corridors.

Louvain Algorithm Results:

Identified 4 distinct habitat clusters.

Modularity Score: 0.6646 (A high score suggests a strong clustering structure).

--- Habitat Cluster Mapping ---

Patch A: Cluster 1

Patch B: Cluster 1

Patch C: Cluster 1

Patch D: Cluster 3

Patch E: Cluster 3

Patch F: Cluster 3

Patch G: Cluster 2

Patch H: Cluster 2

Patch I: Cluster 2

Patch J: Cluster 0

Patch K: Cluster 0

/tmp/ipython-input-742464345.py:69: MatplotlibDeprecationWarning: The get\_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get\_cmap()`` or ``pyplot.get\_cmap()`` instead.

cmap = plt.cm.get\_cmap('viridis', max(partition.values()) + 1)

